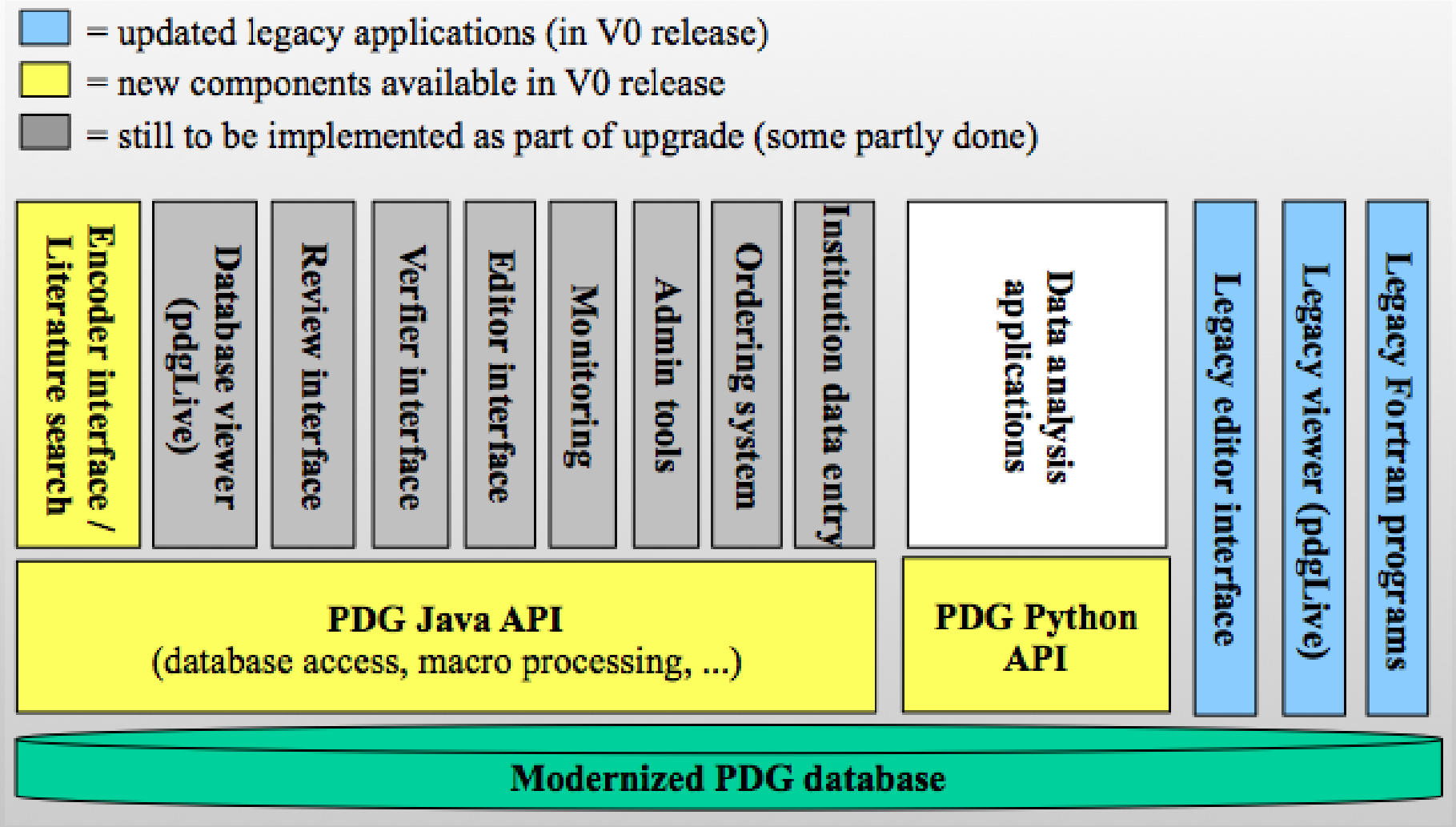
PDG Database

David W. Robertson
Computational Research Division
Lawrence Berkeley National Laboratory

PDG Architecture

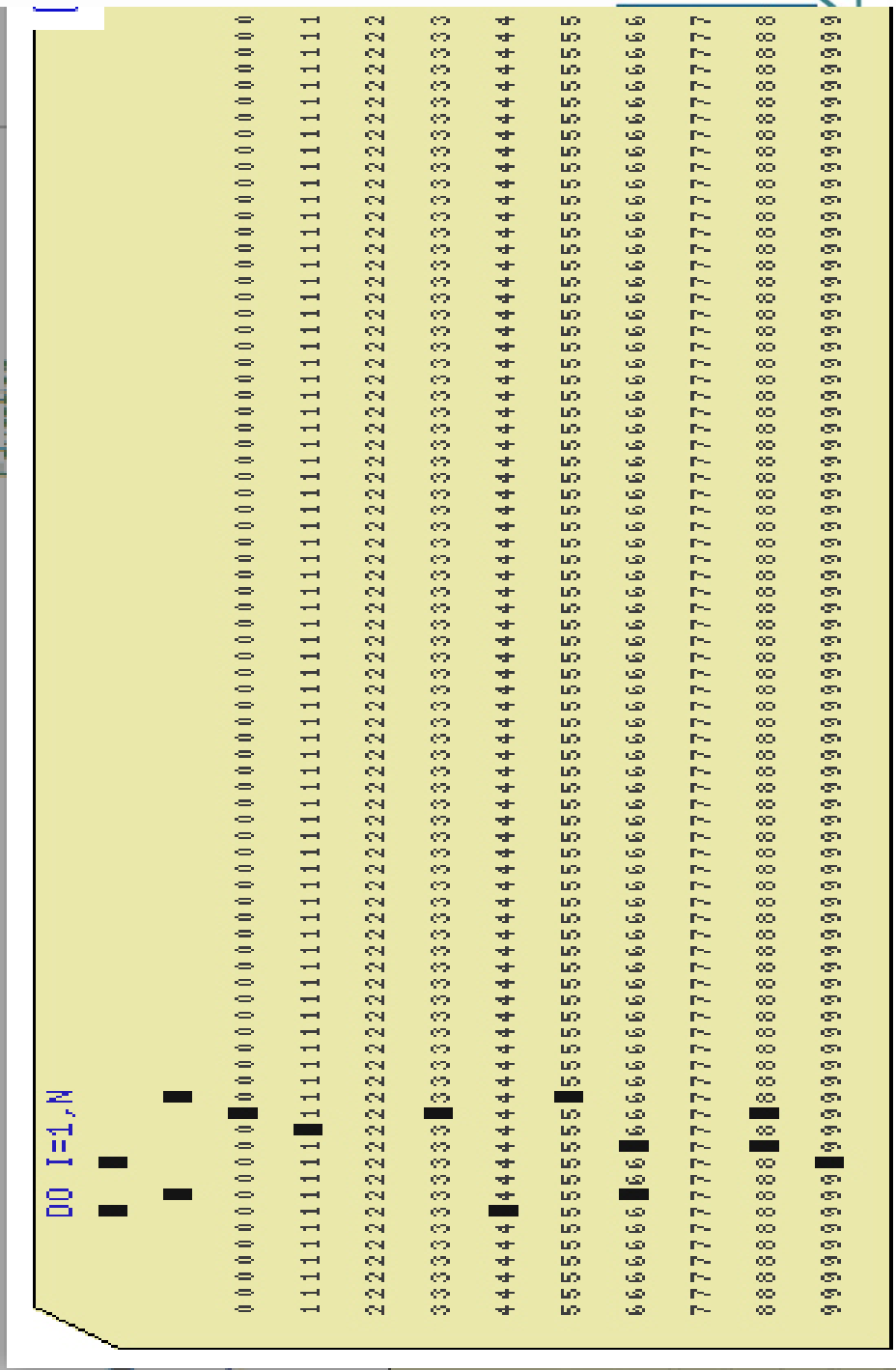


Topics Covered

- **Background**
  - Complex database
  - Why we chose the following goals after careful review of the original system
- **Goals for upgrade**
  - Upgrade database *incrementally*
  - *Modernize* the database, allowing full usage of tools available in Java and Python, and allowing Web-level applications
    - The Web-level applications ensure that it is no longer the editor doing everything, and the process is **scalable**
  - Have a *maintainable* database for years into the future
  - Continue production of the book while under development, and have a *seamless* transition to the upgraded database

History

- **The Review has been produced for *40* years**
  - Originally typewritten text
  - Punch cards in the 1980's
  - Oracle database: 1988-2005
  - PostgreSQL: 2005-present
- **The process has worked for all this time**
  - Adhered to best practices to get the book out
  - The result has been accurate and dependable

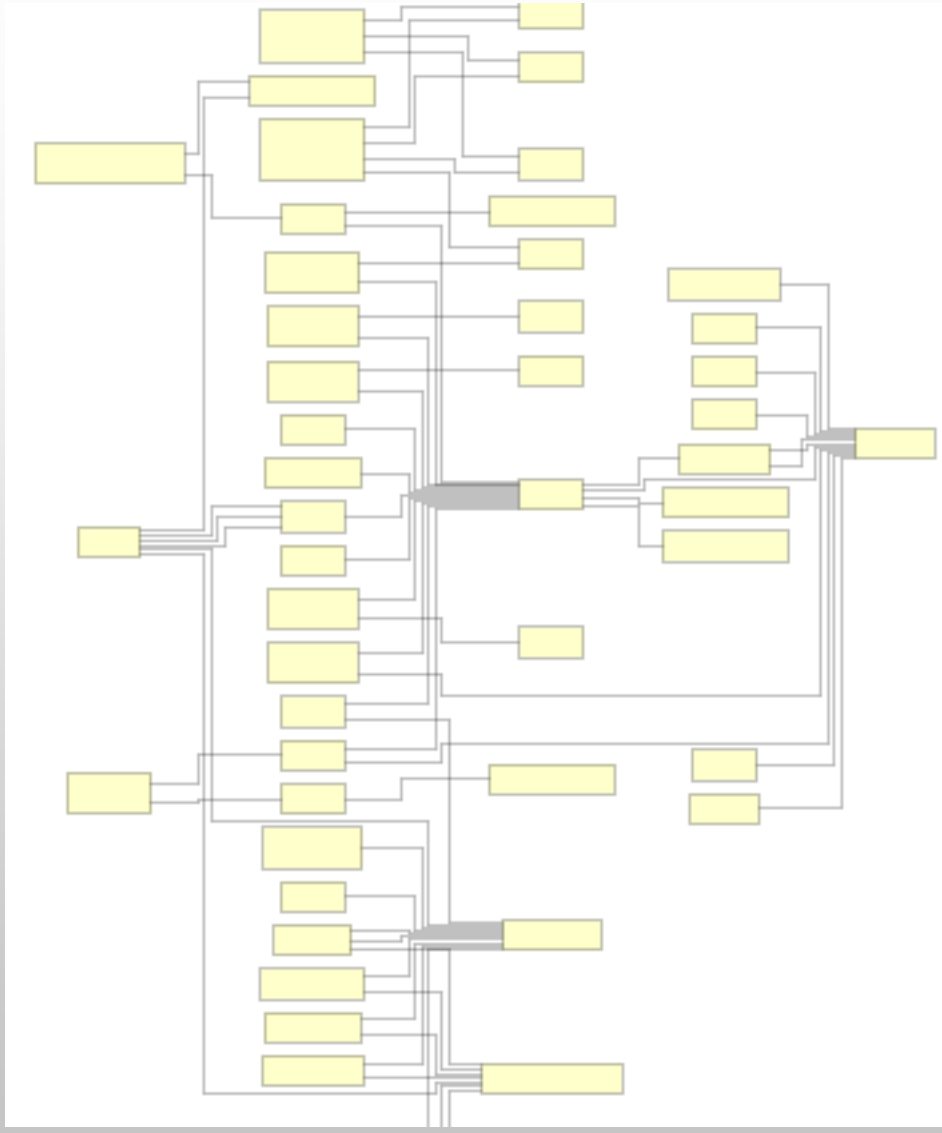


Status before Upgrade

- **Original design worked for many years, but was brittle and required expert knowledge of the database**
- **Complex database with many implicit relationships**
- **Legacy Fortran (110,000 lines) directly accessing the database**
  - Partly carried over into new system; no need to replace
- **Difficult to use with modern database tools**
  - No integrity constraints
    - No primary keys
    - No foreign keys
- **Scalability and maintainability needed to be improved**
  - Assumption of single editor accessing the database: not scalable
  - Documentation was incomplete
  - No task-level change logging

Complex Database

- **38 megabytes, 104 tables, 679 columns**
- **~600,000 rows with many tables having thousands of rows**
- **Multiple relationships between tables, but no constraints**
- **Divided into scientific and book production tables; book production tables refer to scientific tables**



Design Goals

- **Goals were chosen after careful review of the initial status**
- **Changes have to be made *incrementally*; redoing completely not feasible**
  - Decades of effort have gone into a complex database and have to be preserved; complete redesign would have been much larger effort
  - Complete change incompatible with ongoing production of the review
  - Must still maintain compatibility with existing data and existing legacy Fortran programs
- **Move to a *more modern database***
  - Add integrity constraints, which
  - Enables more modern tooling, supporting PostgreSQL multi-user mode in higher level applications using Java and Python
- **Ensure *maintainability* into the future**
- **Have a process to continue production of the review while under development, and ensure a *seamless* transition**

Data Integrity

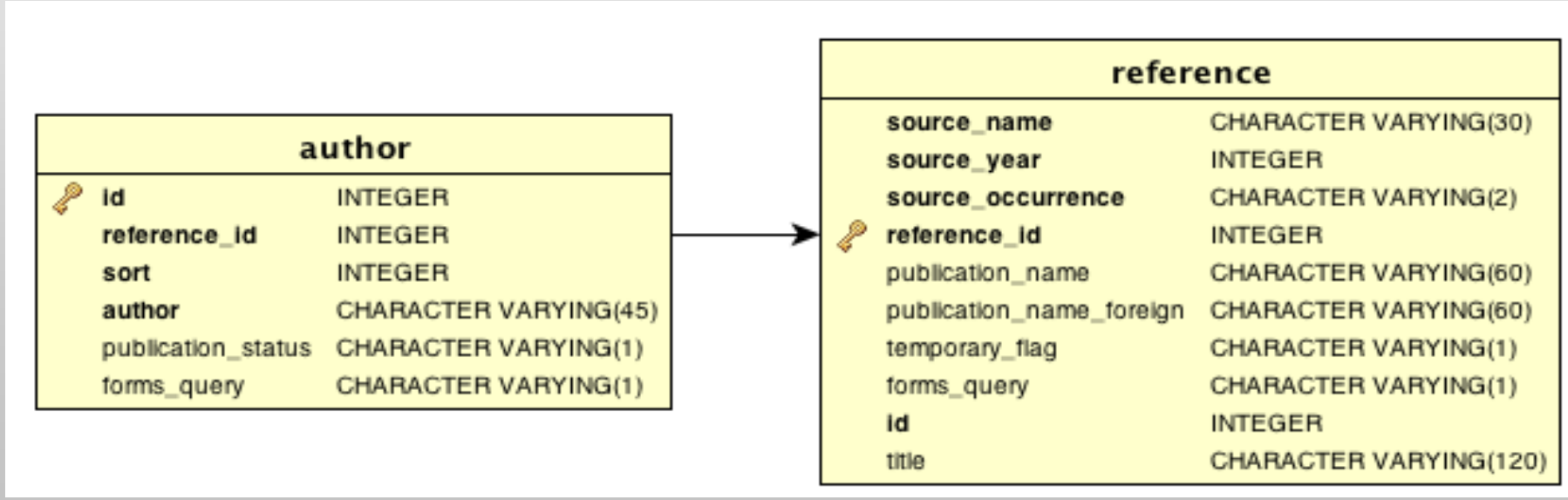
- Provides formal *constraints* to make database consistent and more navigable
- *Primary* keys declared in all tables: entity integrity
- *Foreign* keys declared in many tables: referential integrity

Benefits of Referential Integrity

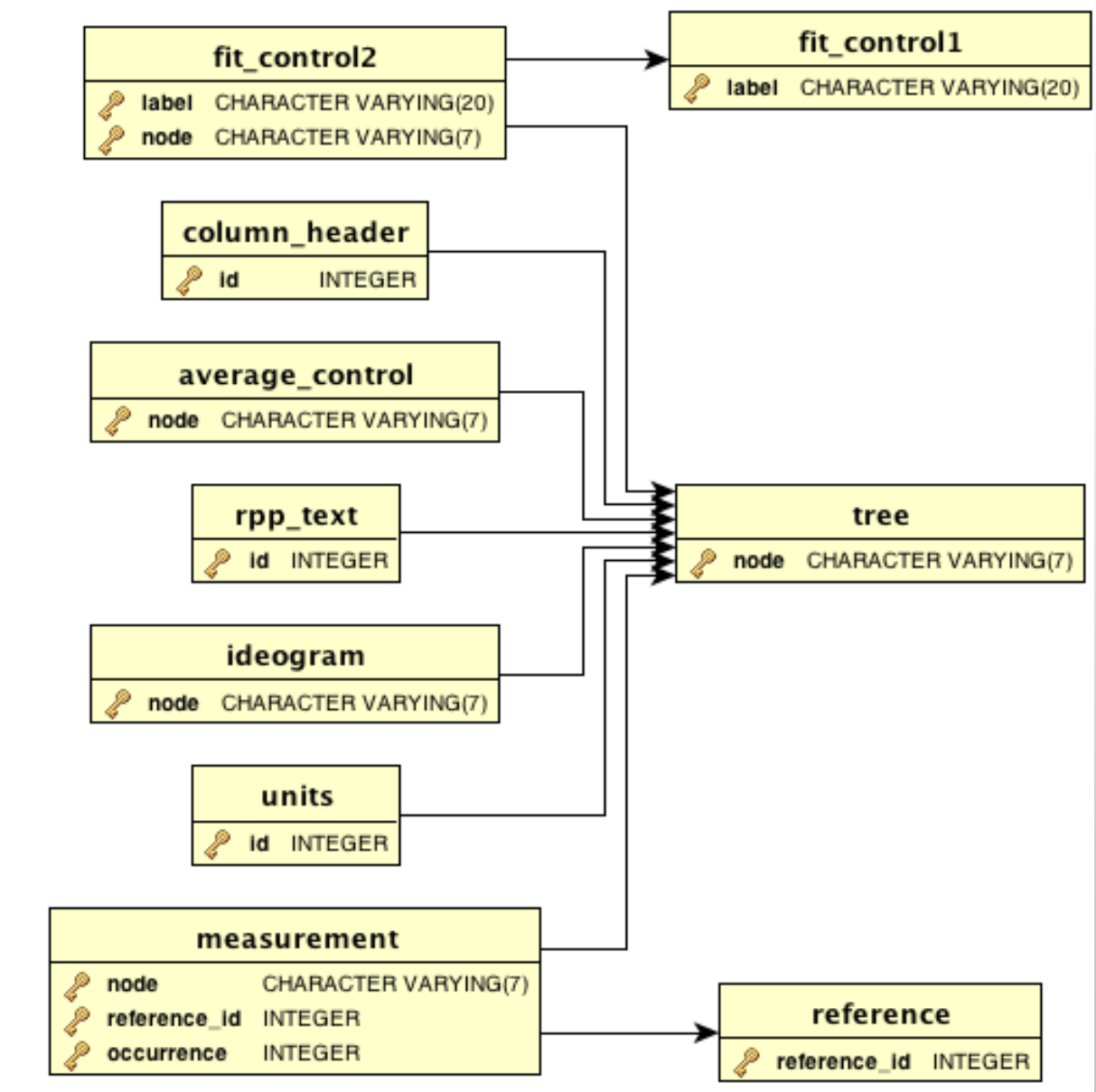
All references by an author can now easily be found.

All authors for a reference are also easily found.

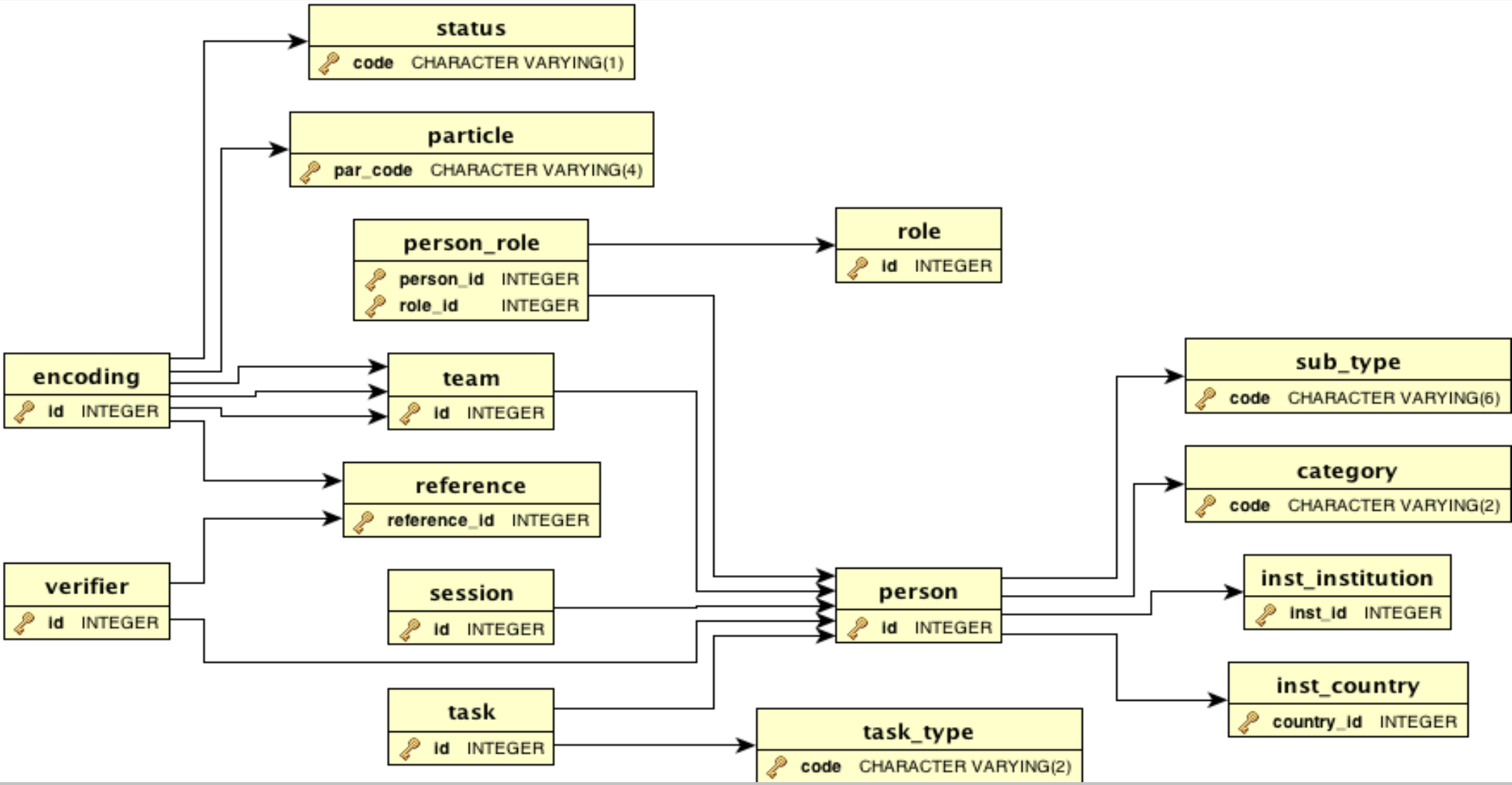
Consistency is *automatically* enforced by the database.



Tree Organization of Review



Workflow Related Tables



Maintainability

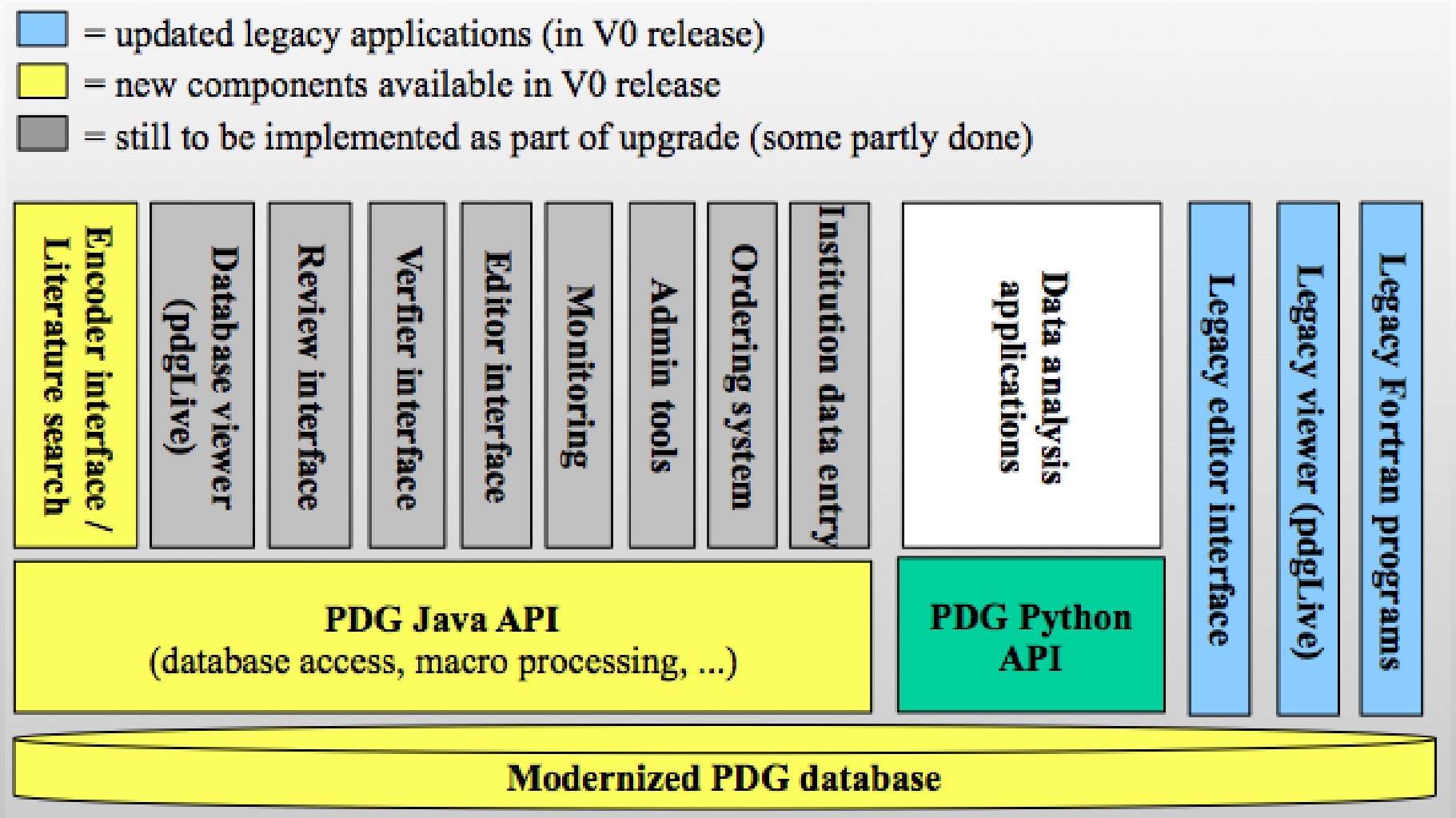
Red Book Documentation

MEASUREMENT			
ID[i]	INT		Unique identifier used for logging changes to the database.
NODE[kf]	CHAR	7	Foreign key to TREE table. Includes a code for the type of measurement being reported.
REFERENCE_ID[kf]	INT	5	The reference containing the reported measurement. Foreign key to REFERENCE table.
OCCURRENCE[k]	INT	1	Usually 1 (which is the default). However, occasionally, one reference will report multiple measurements of the same type, and this field should be used to distinguish such multiple measurements. Furthermore, the value of OCCURRENCE will be used at print time to sort these multiple measurements.
MEASUREMENT	CHAR	150	The value, errors, bounds, irregular data, background estimates, *etc.* as described by the measurement syntax diagrams. At input time, the MEASUREMENT field will be parsed to assure that it conforms to one of a set of acceptable syntaxes. If it does not conform, an error message will be displayed and the user must correct the mistake. For measurements that are to be accepted as is, the field may begin with an asterisk. At print time, MEASUREMENT will again be parsed and typesetting will attempt to line up decimal points, multiply signs, and so on.

Change Logging

- **Transactional logging is opaque, and shows all database operations without context except for time**
- **Task level logging**
  - Ability to see all insertions, updates, and deletions on a *per task* level.
  - Ability to debug mistakes at the task level
- **Chuck's talk has the details**

PDG Architecture



Python API

- **SQL is a very low level way to access the database, and is not programmatic except in a vendor-specific way**
- **A data block in the Review involves a number of tables and relationships**
- **Python API provides interactive API to deal with data block**
  - Uses the *object relational mapping* (ORM) tools SqlAlchemy and SqlSoup for accessing the database
  - ORM tools are important, and Chuck's talk will cover the details
- **Demo showing the simplicity of this API in the afternoon**

Data Block

$B^{\pm}$ MASS

The fit uses m_{B^+}, $(m_{B^0} - m_{B^+})$, and m_{B^0} to determine m_{B^+}, m_{B^0}, and the mass difference.

VALUE (MeV)	EVTS	DOCUMENT ID		TECN	COMMENT
5279.17 ±0.29 OUR FIT					
5279.1 ±0.4 OUR AVERAGE					
5279.10 ±0.41 ±0.36		[1] ACOSTA	06	CDF	$p\overline{p}$ at 1.96 TeV
5279.1 ±0.4 ±0.4	526	[2] CSORNA	00	CLE2	$e^+e^- \to \Upsilon(4S)$
5279.1 ±1.7 ±1.4	147	ABE	96B	CDF	$p\overline{p}$ at 1.8 TeV
• • • We do not use the following data for averages, fits, limits, etc. • • •					
5278.8 ±0.54 ±2.0	362	ALAM	94	CLE2	$e^+e^- \to \Upsilon(4S)$
5278.3 ±0.4 ±2.0		BORTOLETTO	92	CLEO	$e^+e^- \to \Upsilon(4S)$
5280.5 ±1.0 ±2.0		[3] ALBRECHT	90J	ARG	$e^+e^- \to \Upsilon(4S)$
5275.8 ±1.3 ±3.0	32	ALBRECHT	87C	ARG	$e^+e^- \to \Upsilon(4S)$
5278.2 ±1.8 ±3.0	12	[4] ALBRECHT	87D	ARG	$e^+e^- \to \Upsilon(4S)$
5278.6 ±0.8 ±2.0		BEBEK	87	CLEO	$e^+e^- \to \Upsilon(4S)$

[1] Uses exclusively reconstructed final states containing a $J/\psi \to \mu^+\mu^-$ decays.

[2] CSORNA 00 uses fully reconstructed 526 $B^+ \to J/\psi^{(\prime)} K^+$ events and invariant masses without beam constraint.

[3] ALBRECHT 90J assumes 10580 for $\Upsilon(4S)$ mass. Supersedes ALBRECHT 87C and ALBRECHT 87D.

[4] Found using fully reconstructed decays with $J/\psi(1S)$. ALBRECHT 87D assume $m_{\Upsilon(4S)}$ = 10577 MeV.

Upgrade process

Documenting Upgrade Process

Meetings and Communication

- List of action Items
- **Meetings:** Our regular meeting time slot is **weekly** on Thursday from 2pm to 4pm in room 50B-6208. Meetings will be announced or cancelled via our mailing list (see below).
- **Minutes** (where available) can be found on our meeting page.
- **Minireviews:** We will assess the status and progress of different aspects of the project whenever suitable milestones are reached, typically every two to three months. See our minireviews page. Next mini review: assess progress towards v0 deployment, end of March.
- **Mailing lists:** We use the mailman mailing list computing@pdg1.lbl.gov for general e-mail and disucssions related to PDG computing. You can subscribe/unsubscribe to this list at http://pdg1.lbl.gov/mailman/listinfo/computing. An archive of past messages is available at http://pdg1.lbl.gov/mailman/private/computing/.

Computing Upgrade Phase 2 (2008-2011)

- Requirements for the PDG Computing Upgrade
- Use Cases
- Design documents and Design decisions
- Tests and demos
- Development Environment
- Technology Corner - various technologies we're using or planning to use for the upgrade
- PDG Development Database (pdgdev)
- Product Ordering System
- PDG Workspace
- PDG API
- PDG Python API
- PDG Identifiers
- DatabaseMigration
- Setup of new PDG servers
- System administration issues
- Releases
- Database

Upgrade Verification Process

- **A process was put in place to ensure that when the production system was upgraded, everything would work**
- **A number of changes in development database until database frozen; all changes had to be tested**
- **Nightly test procedure**
  - Changes in production database committed as SQL dumps under CVS control
  - Copy of the production database created from SQL dumps, and upgraded database produced by applying SQL scripts
  - Tested the resulting database against the Java and Python API's

Upgrading to V0

- July 2009-May 2010: Ongoing development of upgraded database and Java applications on old machine
- March 2010-May 2010: Database development on machine with room for growth, using PostgreSQL 8.4
- May 2010: Development database schema frozen, Java applications moved to new machine and to PostgreSQL 8.4
- July 2010: All legacy Fortran programs worked with the development PostgreSQL 8.4 database
- August 2010: Production database moved to new machine, and upgraded to incorporate modifications introduced in tested development database
  - V0 release

Current Status

- **Successfully moved from a legacy database to a *modern database***
- **Satisfied constraint of producing the 2010 edition of the review while the modified database was under development**
- **The review produced by the legacy Fortran programs is *identical* using the old and new production database**
- **Database-related work for the remaining interfaces will incorporate the same proven design and verification processes that worked for V0**

Changes Past V0

- **Majority of changes have been accomplished**
  - Additional changes will occur at planned intervals
- **Minor changes still remain**
  - New columns in some tables
  - More foreign keys
- **Implementing remaining interfaces will necessitate new schema but minor changes to existing schema**

Conclusions

- **Database now meets our needs**
  - Shifted constraints from the application to the database level
  - Each new application no longer has to re-implement constraints
  - Database itself now logs every task
- **Deployment was seamless**
- **Well-documented, and maintainable into future**

It is now our production database!!!